

## CHAPTER 7

# BOOLEAN ALGEBRA

The father of Boolean algebra was George Boole, who was an English logician and mathematician. In the spring of 1847, he wrote a pamphlet on symbolic logic. Later he wrote a much larger text on which are founded the mathematical theories of logic. He did not regard logic as a branch of mathematics, but he did point out that a close analogy between symbols of algebra and those symbols which he devised to represent logical forms does exist.

Boolean algebra lay almost dormant until 1937 when Boole's algebra was used to write symbolic analyses of relay and switching circuits. Boolean algebra has now become an important subject to be learned in order to understand electronic computer circuits.

### CLASSES AND ELEMENTS

We have previously determined that in our universe we can logically visualize two divisions; all things of interest in any discussion are in one division, and all other things not of interest are in the other division. These two divisions comprise a set or class called the **UNIVERSAL CLASS**. All objects contained in the universal class are called **ELEMENTS**. We also identify a set or class containing no elements; this class is called the **NULL CLASS**.

If we group some elements of the universal class together to form the combinations which are possible in a particular discussion, we call each of these combinations a class. In Boolean logic, these combinations called classes should not be confused with the null class or universal class. Actually, these classes are subclasses of the universal class. It should also be noted that the elements and classes in Boolean algebra are the sets and subsets previously discussed.

Each class is dependent upon its elements and the possible states (stable, nonstable, or both) that the elements can take.

Boolean algebra is that algebra which is based on Boolean logic and concerned with all elements having only two possible stable states and no unstable states.

To determine the number of classes or combinations of elements in Boolean algebra, we solve for the numerical value of  $2^n$  where  $n$  equals the number of elements. If we have two elements (each element has two possible states) then we have  $2^n$  or  $2^2$  possible classes. If we let the elements be A and B, then A may be true or false and B may be true or false. The classes which could be formed are as follows:

A true and B false  
A true and B true  
A false and B true  
A false and B false

where we use the connective word "and." We could also form classes by use of the connective word "or" which would result in a different form of classes.

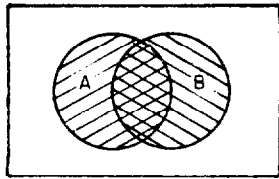
### VENN DIAGRAMS

Since the Venn diagram is a topographical picture of logic, composed of the universal class divided into classes depending on the number of elements, we show this logic as follows.

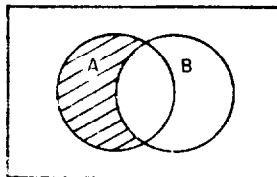
We may consider the universal class as containing submarines and atomic powered sound sources. Let A equal submarines and B equal atomic powered sound sources. Therefore, we have four classes which are:

- (1) Submarines and not atomic
- (2) Submarines and atomic
- (3) Atomic and not submarines
- (4) Not submarines and not atomic

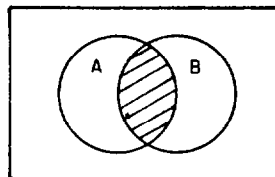
A diagram of these classes is



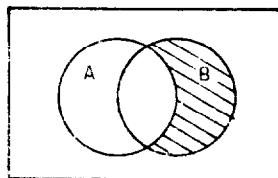
We may show these classes separately by



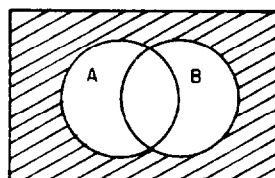
A and not B



A and B

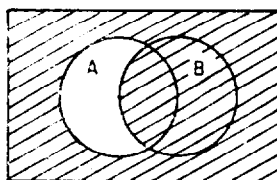


B and not A

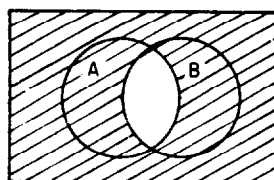


not A and not B

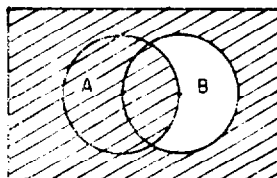
These four classes are called minterms because they represent the four minimum classes. The opposite of the minterms are called maxterms and are shown by



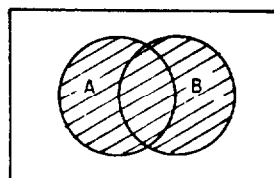
B or not A



not A or not B



A or not B



A or B

We will discuss minterms and maxterms in more detail later in the chapter.

## BASIC EXPRESSIONS

It has been seen that the Venn diagram may be used to represent a picture of logic. The logic previously used was written in longhand, and used the words "and," "or," and "not." We used these words as a basis for combining elements to form classes in Boolean algebra logic descriptions. The symbols from sets and subsets are  $\cap$  for "and,"  $\cup$  for "or," and  $-$  for "not." The relationships of symbols are given by the following:

Sets and subsets	Words	Boolean algebra
$\cap$	and	$\cdot$
$\cup$	or	$+$
$-$	not	$\bar{\phantom{x}}$

The following are examples of these relationships:

- (1)  $A \cdot B$  reads A and B
- (2)  $\bar{A} + B$  reads A or B
- (3)  $\bar{A}$  reads not A

Relationships to the previously indicated classes, about submarines and atomic powered sound, are

- (1) A and not B =  $A \cdot \bar{B}$
- (2) A and B =  $A \cdot B$
- (3) B and not A =  $B \cdot \bar{A}$
- (4) not A and not B =  $\bar{A} \cdot \bar{B}$

also

- (1) B or not A =  $B + \bar{A}$
- (2) not A or not B =  $\bar{A} + \bar{B}$
- (3) A or not B =  $A + \bar{B}$
- (4) A or B =  $A + B$

Notice that

$$\begin{aligned} &A \cdot \bar{B} \\ &A \cdot B \\ &\bar{B} \cdot A \\ &\bar{A} \cdot \bar{B} \end{aligned}$$

are called minterms. As related to algebra, there is a minimum number of terms in each; that is, one. Notice also that

$$\begin{aligned} &\bar{B} + \bar{A} \\ &\bar{A} + \bar{B} \\ &A + \bar{B} \\ &A + B \end{aligned}$$

are called maxterms. As related to algebra, there is a maximum number of terms in each. That is, two.

A further relationship may be made to sets and subsets as follows:

$$\begin{aligned}A \cdot \bar{B} &= A \cap \bar{B} \\A \cdot B &= A \cap B \\B \cdot \bar{A} &= B \cap \bar{A} \\\bar{A} \cdot \bar{B} &= \bar{A} \cap \bar{B}\end{aligned}$$

If we take any of these minterms, such as  $A \cdot B$ , and find its complement we have, according to DeMorgan's theorem

$$\begin{aligned}\overline{(A \cdot B)} &= \overline{(A \cap B)} \\&= \bar{A} \cup \bar{B} \\&= \bar{A} + \bar{B}\end{aligned}$$

which is a maxterm; therefore the complement of a minterm is a maxterm.

### APPLICATIONS TO SWITCHING CIRCUITS

Since Boolean algebra is based upon elements having two possible stable states, it becomes very useful in representing switching circuits. The reason for this is that a switching circuit can be in only one of two possible states. That is, it is either open or it is closed. We may represent these two states as 0 and 1, respectively. Since the binary number system consists of only the symbols 0 and 1, we employ these symbols in Boolean algebra and call this "binary Boolean algebra."

#### THE "AND" OPERATION

Let us consider the Venn diagram in figure 7-1 (A). Its classes are labeled using the basic expressions of Boolean algebra. Note that there are two elements, or variables, A and B. The shaded area represents the class of elements that are  $A \cdot B$  in Boolean notation and is expressed as:

$$f(A,B) = A \cdot B$$

The other three classes are also indicated in figure 7-1 (A). This expression is called an AND operation because it represents one of the four minterms previously discussed. Recall that AND indicates class intersection and both A and B must be considered simultaneously.

We can conclude then that a minterm of n variables is a logical product of these n variables with each variable present in either its noncomplemented or its complemented form, and is considered an AND operation.

For any Boolean function there is a corresponding truth table which shows, in tabular form, the true conditions of the function for each way in which conditions can be assigned its variables. In Boolean algebra, 0 and 1 are the symbols assigned to the variables of any function. Figure 7-1 (B) shows the AND operation function of two variables and its corresponding truth table.

This function can be seen to be true if one thinks of the logic involved: AB is equal to A and B which is the function  $f(A,B)$ . Thus, if either A or B takes the condition of 0, or both take this condition, then the function  $f(A,B)$  equal AB is equal to 0. But if both A and B take the condition of 1 then the AND operation function has the condition of 1.

Figure 7-1 (C) shows a switching circuit for the function  $f(A,B)$  equal AB in that there will be an output only if both A and B are closed. An output in this case equals 1. If either switch is open, 0 condition, then there will be no output or 0.

In any digital computer equipment, there will be many circuits like the one shown in figure 7-1 (C). In order to analyze circuit operation, it is necessary to refer frequently to these circuits without looking at their switch arrangements. This is done by logic diagram mechanization as shown in figure 7-1 (D). This indicates that there are two inputs, A and B, into an AND operation circuit producing the function in Boolean algebra form of AB. These diagrams simplify equipment circuit diagrams by indicating operations without drawing all the circuit details.

It should be understood that while the previous discussion concerning the AND operation dealt with only two variables that any number of variables will fit the discussion. For example, in figure 7-2 three variables are shown along with their Venn diagram, truth table, switching circuit, and logic diagram mechanization.

#### THE "OR" OPERATION

We will now consider the Venn diagram in figure 7-3 (A). Note that there are two elements, or variables, A and B. The shaded area represents the class of elements that are  $A + B$

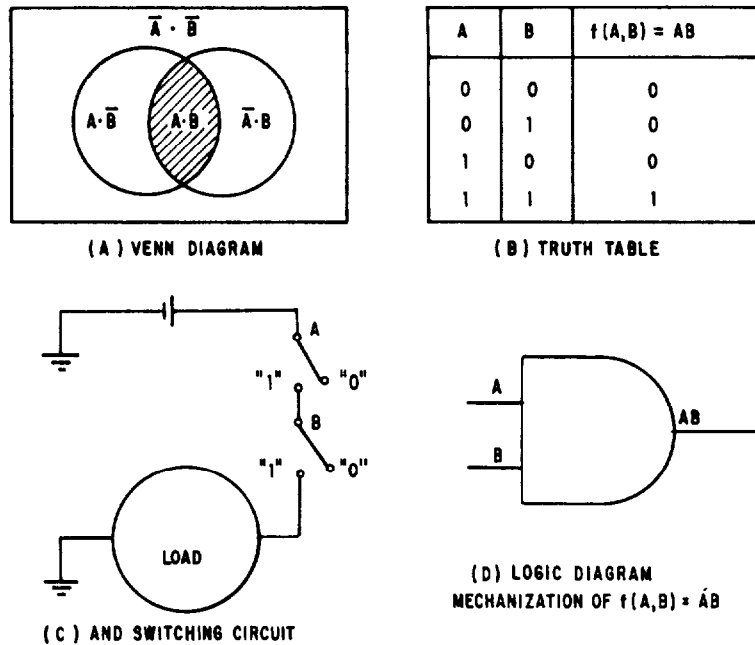


Figure 7-1.—The AND operation.

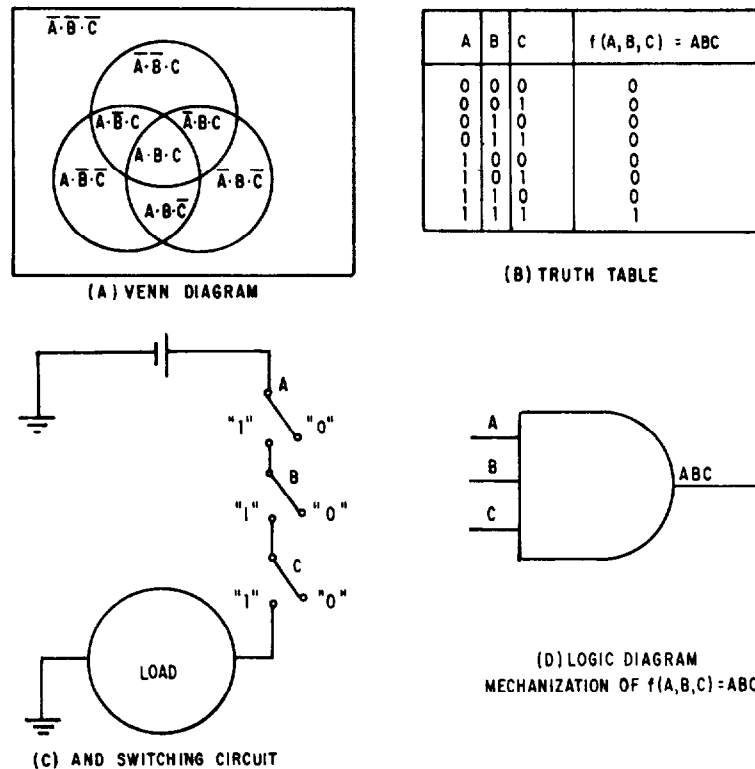


Figure 7-2.—The AND operation (three variables).

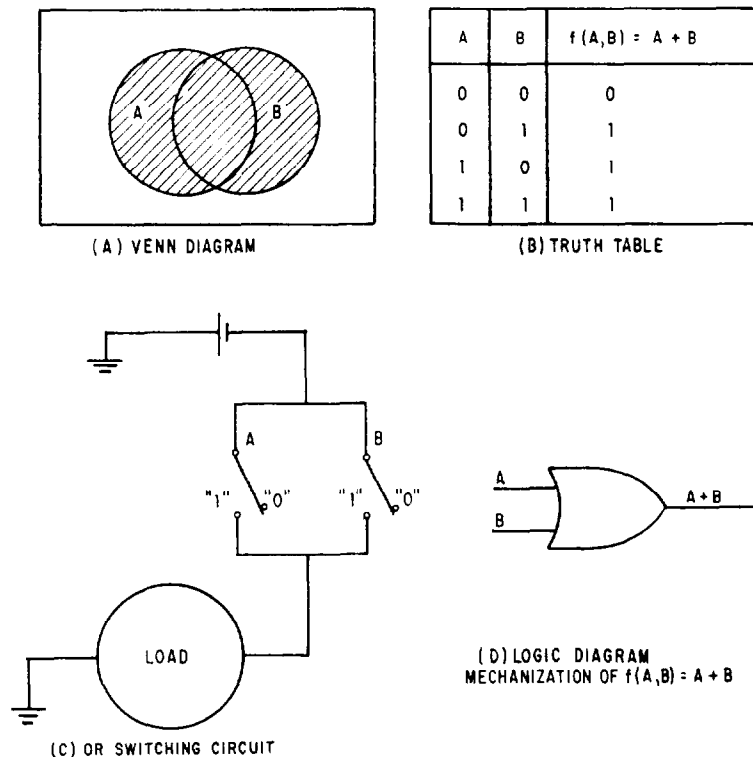


Figure 7-3.—The OR operation.

in Boolean notation and is expressed in Boolean algebra as:

$$f(A,B) = A + B$$

This expression is called an OR operation for it represents one of the four maxterms previously discussed. Recall that OR indicates class union and either A or B or both must be considered.

We can conclude then that a maxterm of n variables is a logical sum of these n variables where each variable is present in either its noncomplemented or its complemented form.

In figure 7-3 (B) the truth table of an OR operation is shown. This truth table can be seen to be true if one thinks of  $A + B$  being equal to A or B which is the function  $f(A,B)$ . Thus if A or B takes the value 1, then  $f(A,B)$  must equal 1. If not, then the function equals zero.

Figure 7-3 (C) shows a switching circuit for the OR operation which is two or more switches

in parallel. It is apparent that the circuit will transmit if either A or B is in a closed position; that is, equal to 1. If, and only if, both A and B are open, equal to 0, the circuit will not transmit.

The logic diagram for the OR operation is given in figure 7-3 (D). This means that there are two inputs, A and B, into an OR operation circuit producing the function in Boolean form of  $A + B$ . Note the difference in the diagram from that of figure 7-2 (D).

As in the discussion of the AND operation the OR operation may also be used with more than two inputs. Figure 7-4 shows the OR operation with three inputs.

#### THE "NOT" OPERATION

The shaded area in figure 7-5 (A) represents the complement of A which in Boolean algebra is  $\bar{A}$  and read as "NOT A." The expression  $f(A)$  equals  $\bar{A}$  is called a NOT operation. The

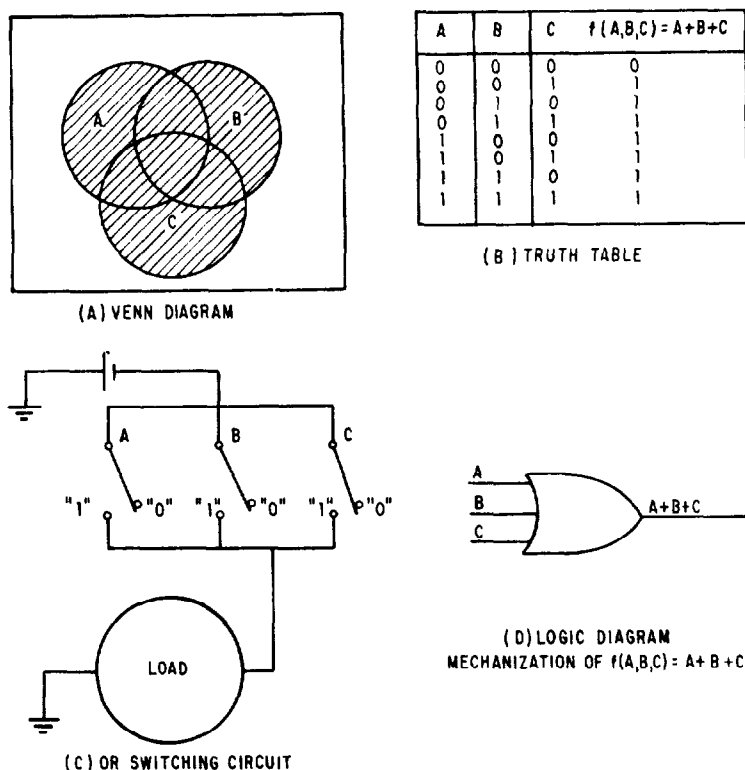


Figure 7-4.—The OR operation (three variables).

truth table for the NOT operation is explained by the NOT switching circuit. The requirement of a NOT circuit is that a signal injected at the input produce the complement of this signal at the output. Thus, in figure 7-5 (C) it can be seen that when switch A is closed, that is, equal to 1, the relay opens the circuit to the load. When switch A is open, that is, equal to 0, the relay completes a closed circuit to the load. The logic diagram for the NOT operation is given in figure 7-5 (D). This means that A is the input to a NOT operation circuit and gives an output of  $\bar{A}$ . The NOT operation may be applied to any operation circuit such as AND or OR. This is discussed in the following section.

#### THE "NOR" OPERATION

The shaded area in figure 7-6 (A) represents the quantity, A OR B, negated. If reference is made to the preceding chapter it will be found that this figure is identical to the minterm

expression  $\bar{A}\bar{B}$ ; that is, A OR B negated is  $\bar{A}\bar{B}$  and by application of DeMorgan's theorem is equal to  $\bar{A+B}$ .

The truth table for the NOR operation is shown in figure 7-6 (B). The table shows that if either A or B is equal to 1, then  $f(A,B)$  is equal to 0. Furthermore, if A and B equal 0, then  $f(A,B)$  equals 1.

The NOR operation is a combination of the OR operation and the NOT operation. The NOR switching circuit in figure 7-6 (C) is the OR circuit placed in series with the NOT circuit. If either switch A, switch B, or both are in the closed position, equal to 1, then there is no transmission to the load. If both switches A and B are open, equal to 0, then current is transmitted to the load.

The logic diagram mechanization of  $f(A,B)$  equal  $\bar{A+B}$  (NOR operation) is shown in figure 7-6 (D). It uses both the OR logic diagrams and the NOT logic diagrams. The NOR logic diagram mechanization shows there are two

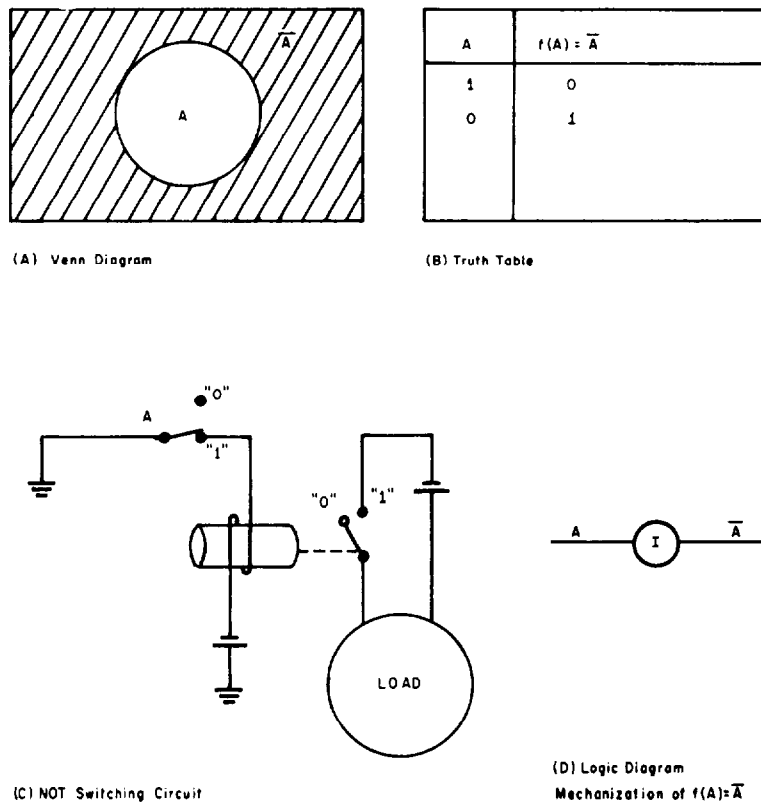


Figure 7-5.—The NOT operation.

inputs, A and B, into an OR circuit producing the function in Boolean form of  $A+B$ . This function is the input to the NOT (inverter) which gives the output, in Boolean form, of  $\overline{A+B}$ . Note that the whole quantity of  $A+B$  is complemented and not the separate variables.

#### THE "NAND" OPERATION

The shaded area in figure 7-7 (A) represents the quantity A AND B negated (NOT), and is a maxterm expression. Notice that  $\overline{AB}$  is equal to the maxterm expression  $\bar{A} + \bar{B}$ .

The truth table is shown for the NAND operation in figure 7-7 (B). When A and B equal 1, then  $f(A,B)$  is equal to 0. In all other cases, the function is equal to 1.

The NAND operation is a combination of the AND operation and the NOT operation. The NAND switching circuit in figure 7-7 (C) is the AND circuit put in series with the NOT circuit.

If either switch A or B is open, equal to 0, then current is transmitted to the load. If both switch A and B are closed, equal to 1, then there is no transmission to the load.

The logic diagram mechanization of  $f(A,B)$  equal  $\overline{AB}$  (NAND operation) is shown in figure 7-7 (D). The AND operation logic diagram and the NOT logic diagram mechanization shows that there are two inputs, A and B, into the AND circuit producing the function in Boolean form of  $AB$ . This function is the input to the NOT circuit which gives the output, in Boolean form, of  $\overline{AB}$ . Note that the entire quantity  $AB$  is complemented and not the separate variables.

It should be noted that in the previously discussed logic diagrams that each input signal represents the operation of a switch, circuit, or other component part.

Generally, a Boolean expression that has been inverted is said to be NOTTED. While we have previously used the inverter symbol

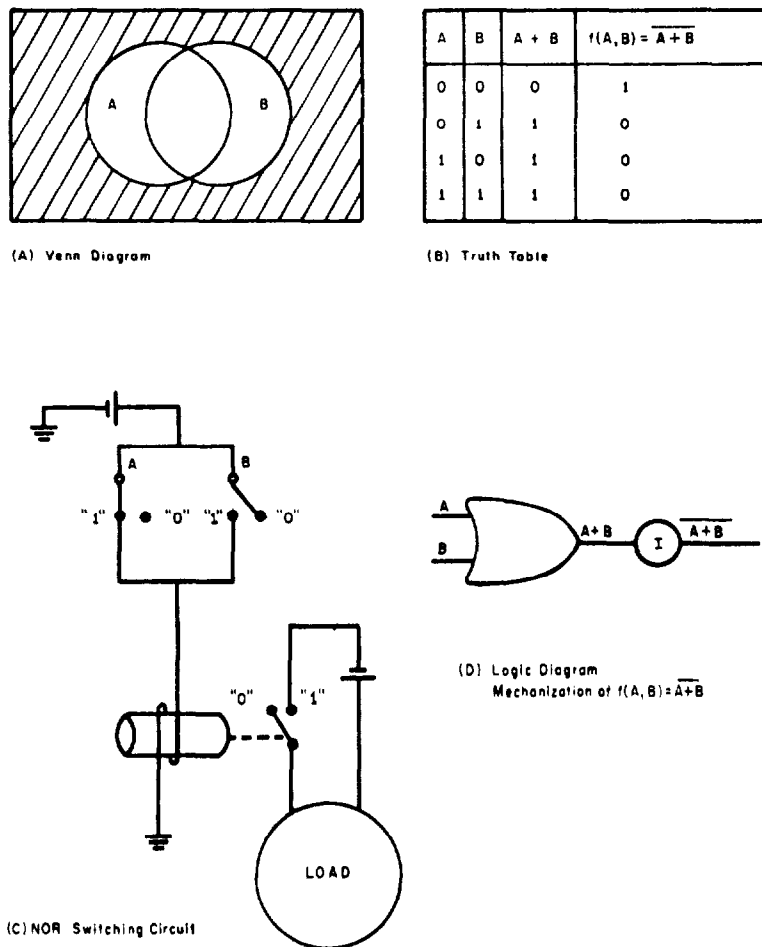


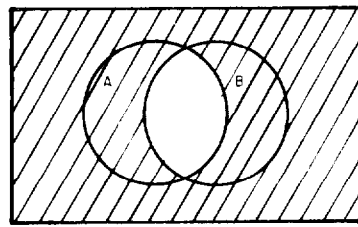
Figure 7-6.—The NOR operation.

separate from the AND or OR logic diagram it is common practice to show the NAND or NOR logic diagrams as indicated in figure 7-8, in accordance with American Standard for Graphic Symbols for Logic Diagrams (ASA Y32.14-1962).

The output of a NAND or a NOR gate is a NOTTED expression. The vinculum is used to indicate that such an expression has been NOTTED. Therefore, the output of a NAND gate having inputs A,B will appear as  $\overline{AB}$  and the output of a NOR gate having inputs A,B will appear as  $\overline{A+B}$ . If any of the inputs to a logic gate are themselves NOTTED a vinculum will appear over the letter representing an input. Examples are shown in figure 7-8.

#### OUTPUT USED AS INPUT

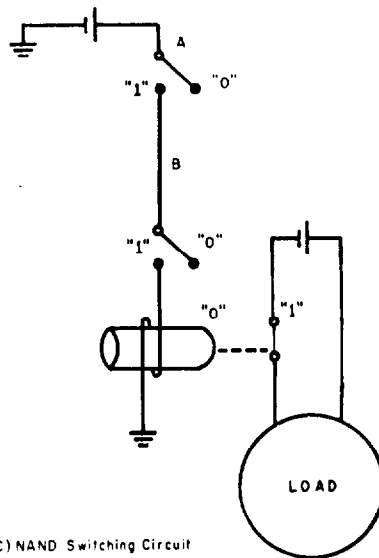
The output from one gate may be an input to another gate. If so, that input will contain two or more letters. Figure 7-9 (A) shows an OR gate feeding into an OR gate. There are four possible combinations of inputs and logic symbols. These are shown in figure 7-9 (B,C,D,E). Notice that signs of grouping occur in all outputs except the AND input to the OR gate. The AB, in this case, is naturally grouped because the letters are written together and are separated from C by the OR sign. Figure 7-10 shows several different cases along with the proper output expressions.



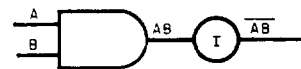
(A) Venn Diagram

A	B	AB	$f(A,B) = \overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

(B) Truth Table



(C) NAND Switching Circuit



(D) Logic Diagram  
Mechanization of  $f(A,B) = \overline{AB}$

Figure 7-7.—The NAND operation.

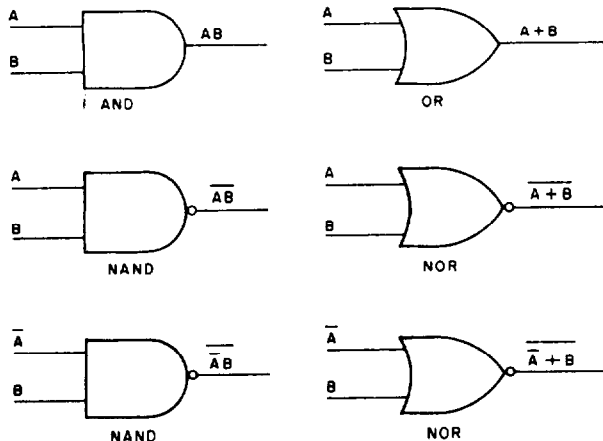


Figure 7-8.—American Standard Logic Symbols.

Although the vinculum is not used in place of parentheses or brackets, it is also a grouping sign. Consider the NOR symbol of figure 7-11 (A). The  $AB$  and  $C$  are the inputs to the OR circuit and form  $AB + C$ . The  $AB + C$  is then inverted to form  $\overline{AB + C}$ . The vinculum groups whatever portion or portions of the output expression that has been inverted. Figure 7-11 (B,C,D) gives examples of this type output.

To determine the output of a logic diagram, find the output of each logic symbol in the diagram. You should begin with the inputs at the left and move right, using the output of each logic symbol as an input to the following symbol, as illustrated in figure 7-12.

When determining the output of a logic diagram, one should be careful of the two most

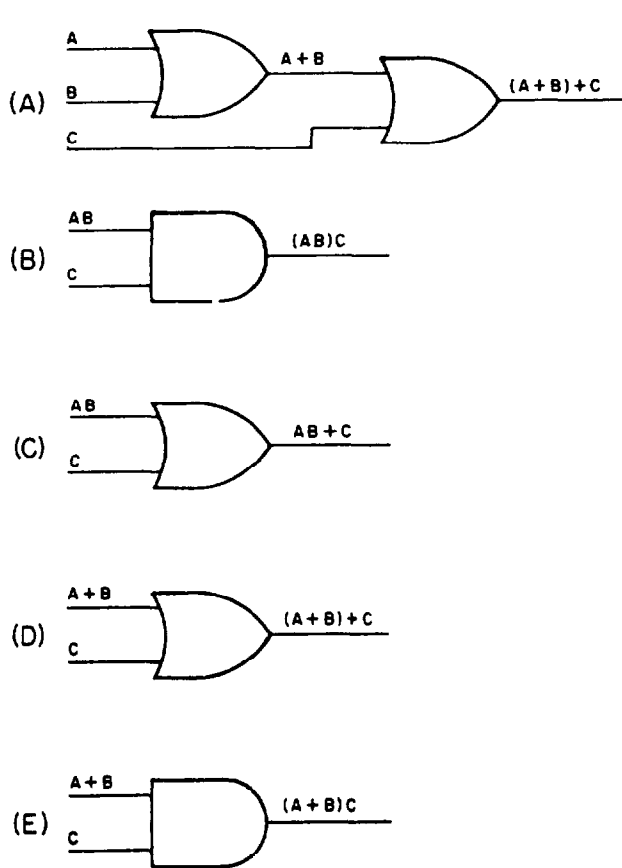
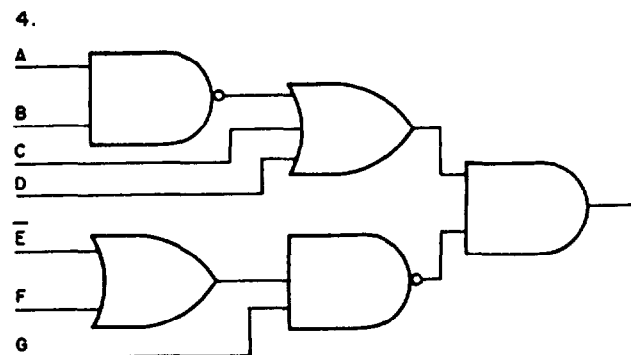
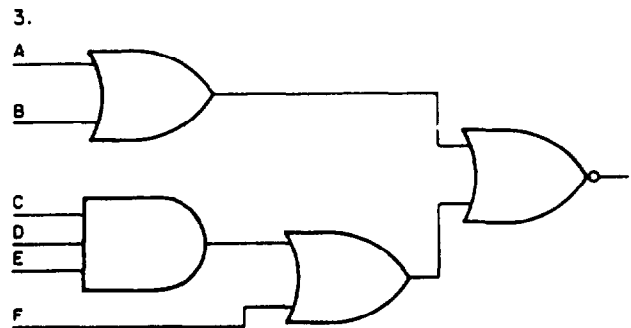
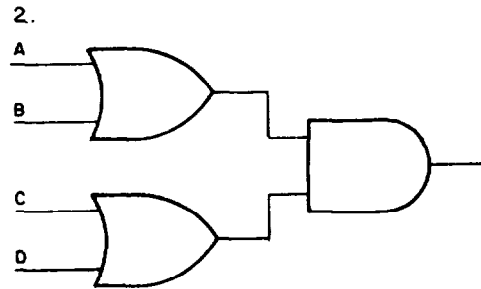
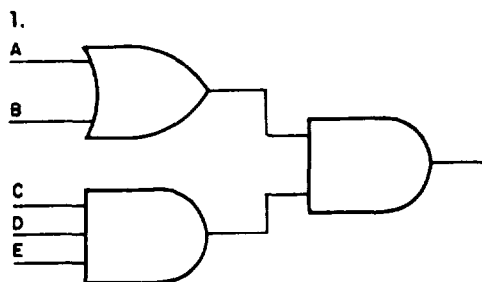


Figure 7-9.—Output as an input.

common mistakes which are leaving out vincula and leaving out grouping signs.

**PROBLEMS:** Find the outputs of the following logic diagrams.



**ANSWERS:**

1.  $(A + B)(CDE)$
2.  $(A + B)(C + D)$
3.  $\overline{(A + B) + (CDE + F)}$
4.  $(\overline{AB} + C + D)[G(\overline{E} + F)]$

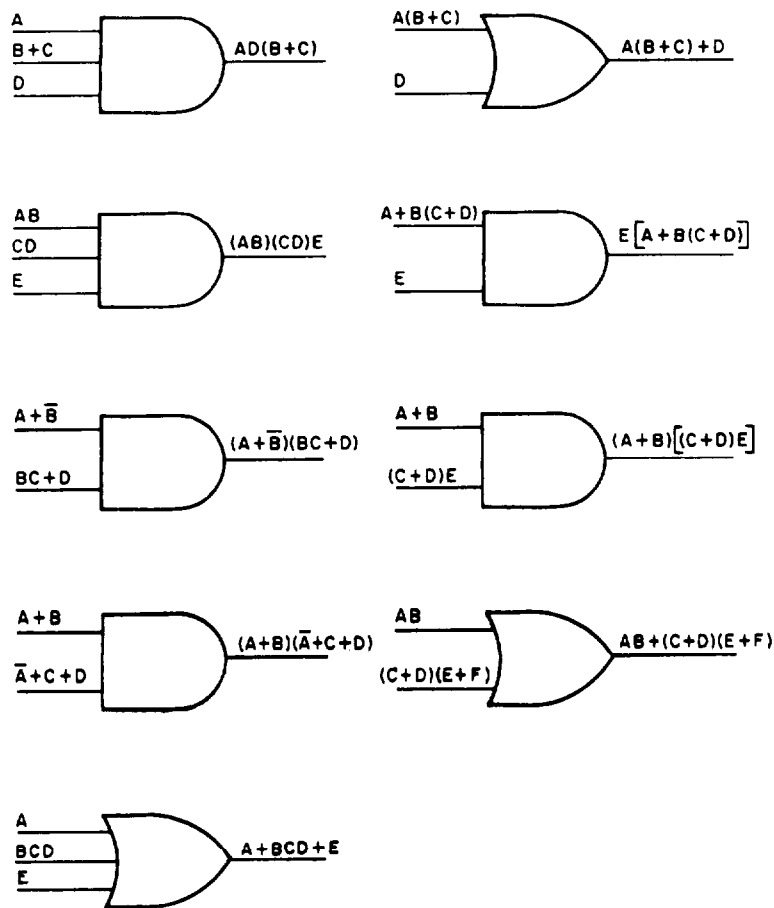


Figure 7-10.—Examples of grouping.

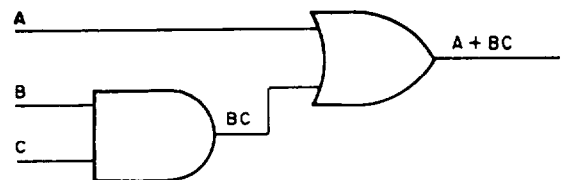
### DEDUCING INPUTS FROM OUTPUTS

In order to draw a logic diagram from an output expression you should start with the output and work toward the input. Separate, in steps, the output expression until you have all single-letter inputs. If letters are grouped, first separate the group from other groups or letters, then separate the letters within groups.

To diagram the input that produces  $A + BC$ , you would first separate  $A$  from  $BC$  by using an OR logic symbol; that is,



You now draw an AND logic symbol to separate  $B$  from  $C$ , and extend all lines to a common column on the left. This is shown by the following diagram.



One common mistake in drawing the simplest possible diagram from an output expression is

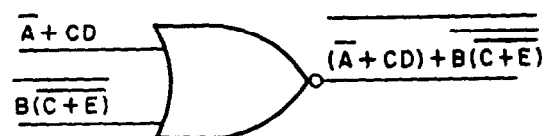
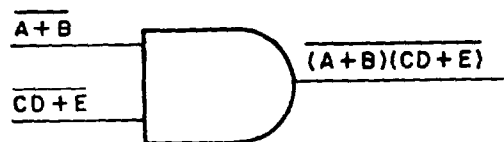
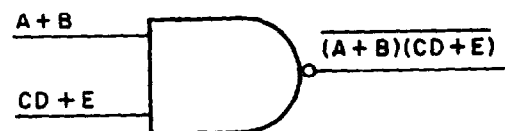
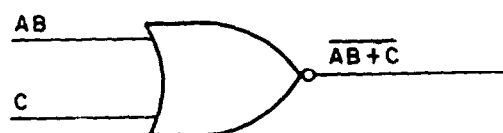
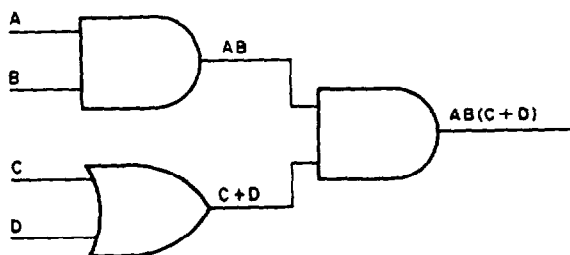
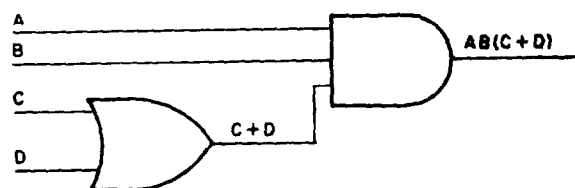


Figure 7-11.—Vinculum as grouping sign.

when the expression is similar to  $AB(C + D)$ . The mistake is made by drawing

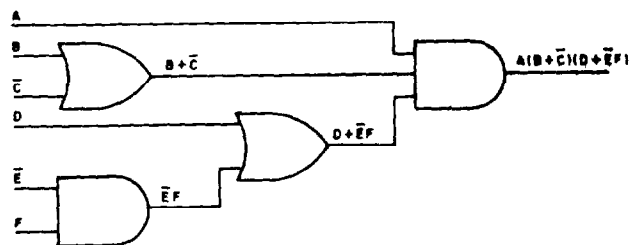


If the foregoing were your results, you would have failed to notice that A, B, and (C + D) were all ANDed together. You should have drawn



which would have saved the use of one gate. A gate is considered one circuit such as OR, AND, NOR, etc.

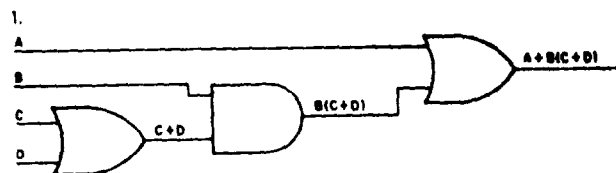
To diagram the expression  $A(B + \bar{C})(D + \bar{E}F)$  write



**PROBLEMS:** Draw the logic diagrams for the following expressions.

1.  $A + B(C + D)$
2.  $(A + B + C)D + E$
3.  $(A + B + C)DE$
4.  $ABC(D + E)$

**ANSWERS:**



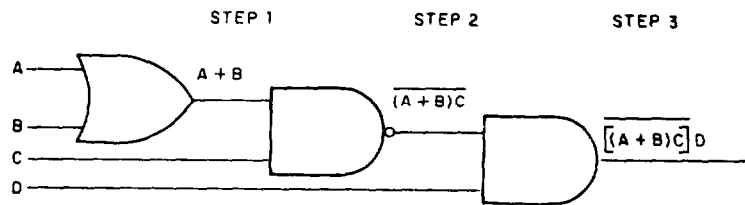
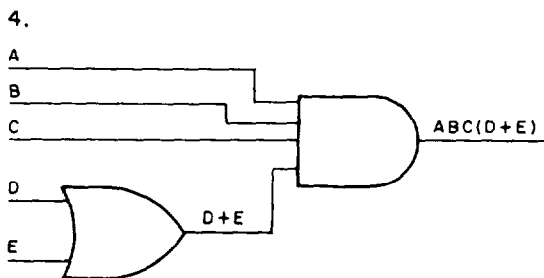
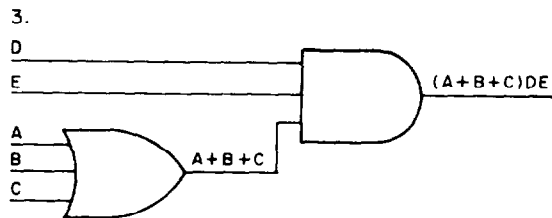
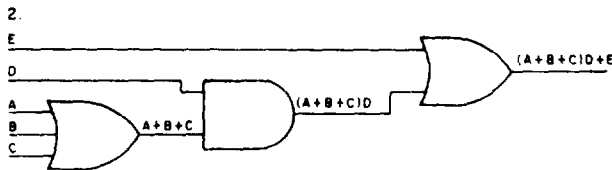


Figure 7-12.--Steps for determining output.



The laws of Boolean algebra may also be used for trouble-shooting defective components or for locating errors in computer programs. It should be understood that not all of the laws are similar to the laws of ordinary algebra.

### LAW OF IDENTITY

This law is shown as

$$A = A$$

$$\bar{A} = \bar{A}$$

and indicates that any letter, number, or expression is equal to itself. The law of identity is shown in figure 7-13.

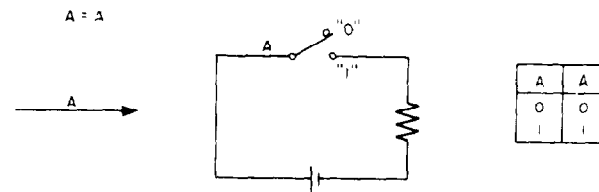


Figure 7-13.--Law of Identity.

### POSTULATES AND THEOREMS

In this section we will discuss the basic laws of Boolean algebra which enables one to simplify many Boolean expressions. By applying the basic laws, the digital systems designer can be sure a circuit is in the simplest possible algebraic form. The actual application of the basic laws will be discussed in the following chapter.

### COMMUTATIVE LAW

The commutative law is:

$$AB = BA$$

and

$$A + B = B + A$$

which is shown in figure 7-14. This indicates that when inputs to a logic symbol are ANDed

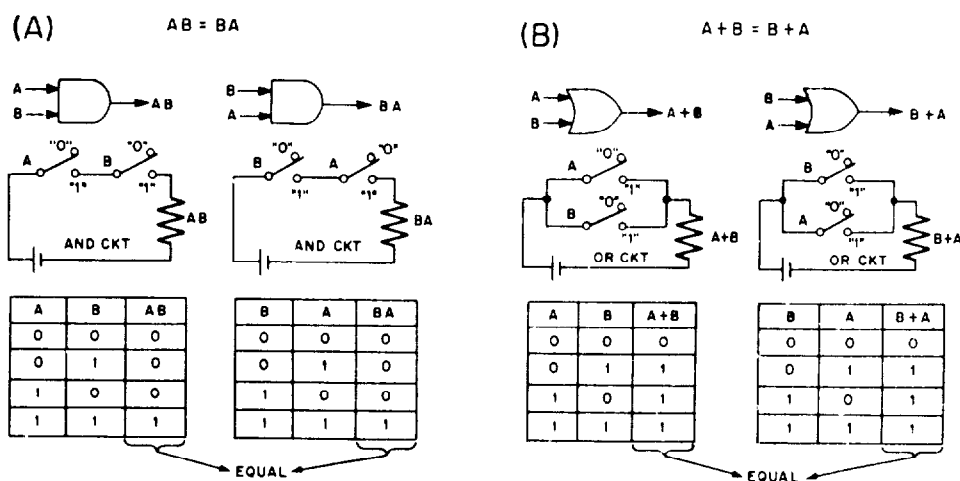


Figure 7-14.—Commutative Law.

or ORed, the order in which they are written does not affect the binary value of the output; that is,

$$R(S + T) = (S + T)R$$

and

$$A(BC + D + E) = (E + BC + D)A$$

### ASSOCIATIVE LAW

The associative law is:

$$A(BC) = (AB)C$$

and

$$A + (B + C) = (A + B) + C$$

which is shown in figure 7-15. This indicates that when inputs to a logic symbol are ANDed or ORed, the order in which they are grouped does not affect the binary value of the output; that is,

$$ABC + D(EF) = (AB)C + DEF$$

and

$$C + (D + E) + (F + G) = C + D + E + F + G$$

### IDEMPOTENT LAW

As seen in figure 7-16, if A is ANDed with A or if A is ORed with A, the output will equal A; that is,

$$AA = A$$

$$A + A = A$$

and

$$(RS)(RS) = RS$$

### LAW OF DOUBLE NEGATION

This law is:

$$\overline{\overline{A}} = A$$

which indicates that when two bars of equal length cover the same letter or expression, both may be removed. This is shown in figure 7-17. Examples are

$$\overline{\overline{AB}} = AB$$

and

$$\overline{\overline{AB}} + \overline{\overline{X}} = \overline{AB} + \overline{X}$$

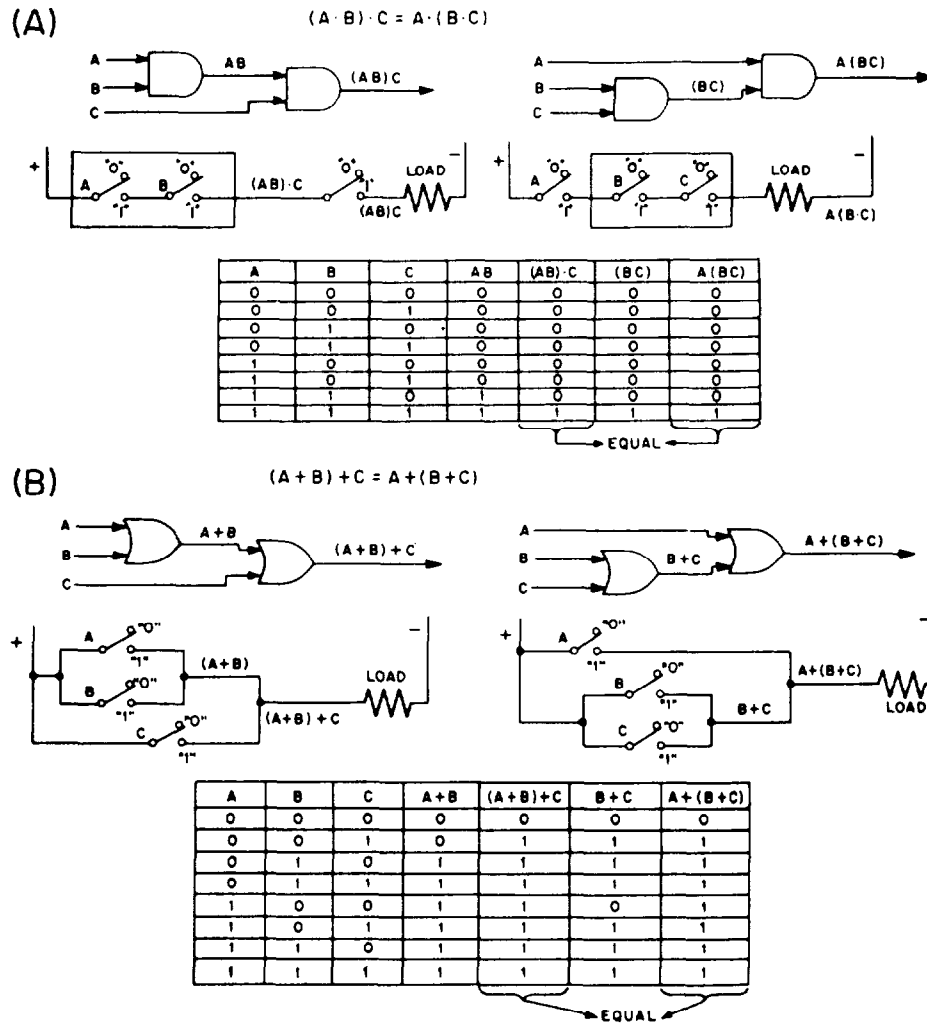


Figure 7-15.—Associative Law.

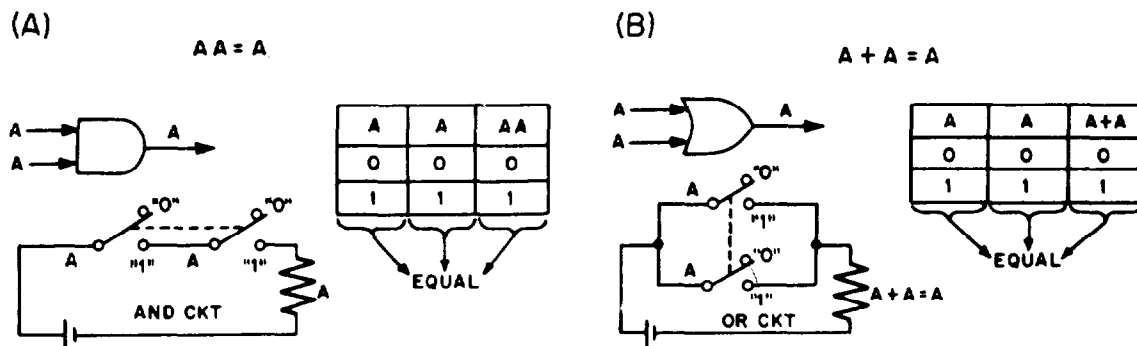


Figure 7-16.—Idempotent Law.

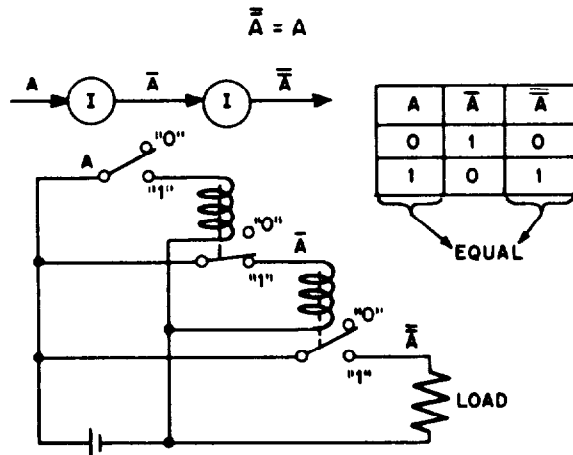


Figure 7-17.—Law of Double Negation.

### COMPLEMENTARY LAW

This law is stated as:

$$A\bar{A} = 0$$

and

$$A + \bar{A} = 1$$

which indicates that when any letter or expression is ANDed with its complement, the output is 0. Also, when any letter or expression is ORed with its complement, the output is 1. This is shown in figure 7-18. Examples are:

$$CD\bar{C}\bar{D} = 0$$

and

$$\overline{ABC} + ABC = 1$$

### LAW OF INTERSECTION

As shown in figure 7-19, if one input to an AND circuit has a value of 1 the output will take the value of the other input. That is, if the two inputs to an AND circuit are 1 and A, then when A is 1 the output will be 1 and when A is 0 the output will be 0. If the inputs are 0 and A, then the output will always be 0.

The law of intersection is given by the following:

$$A \cdot 1 = A$$

and

$$A \cdot 0 = 0$$

Examples are:

$$AB \cdot 1 = AB$$

and

$$CD \cdot 0 = 0$$

### LAW OF UNION

As shown in figure 7-20, if one input to an OR circuit has a binary value of 1, the output will be 1. If the inputs are 0 and A, the output will be the same as the value of A.

The law of union is given by the following:

$$A + 1 = 1$$

and

$$A + 0 = A$$

Examples of this law are as follows:

$$1 + ABC = 1$$

and

$$E + 0(AB) = E$$

### LAW OF DUALIZATION (DeMorgan's Theorem)

To split a vinculum that extends over more than one letter, and to join separate vincula into one vinculum requires the use of the law of dualization. This law is commonly referred to as DeMorgan's theorem. This law is shown in figure 7-21.

DeMorgan's theorem may be written as follows:

$$\overline{AB} = \bar{A} + \bar{B}$$

and

$$\overline{A + B} = \bar{A} \bar{B}$$

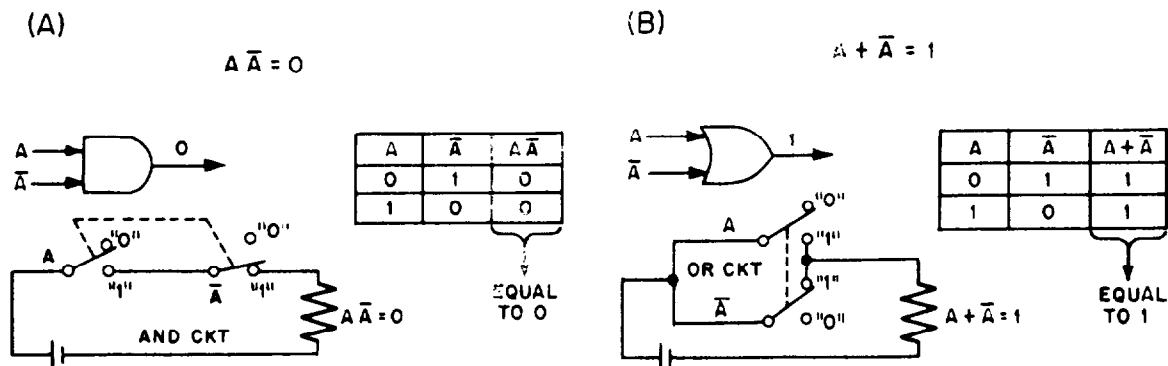


Figure 7-18.—Complementary Law.

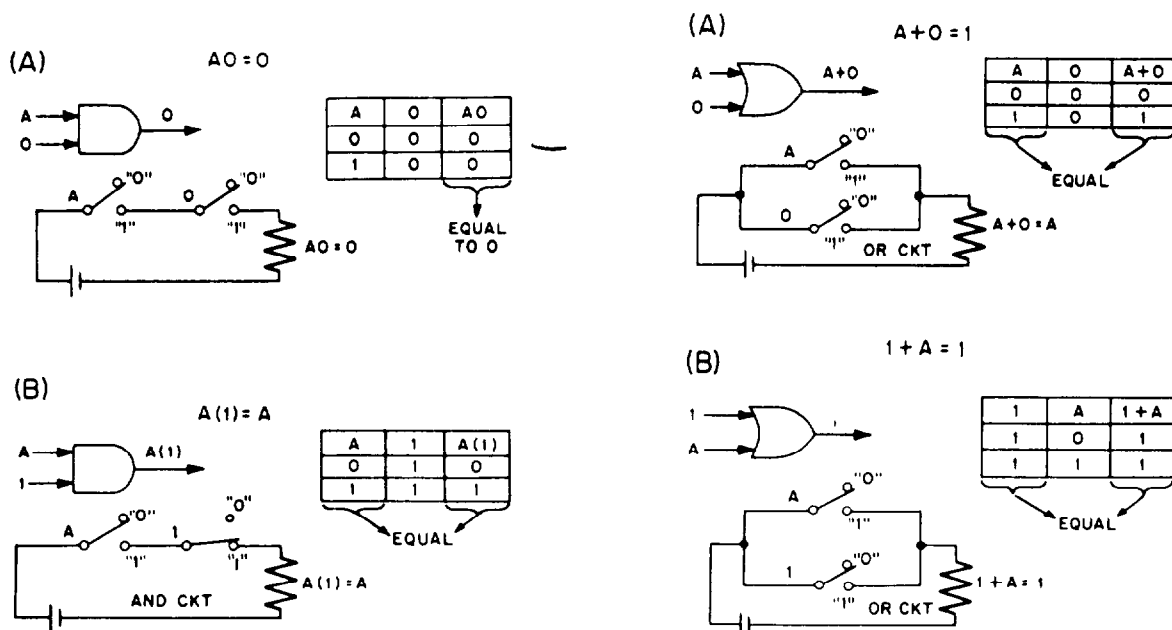


Figure 7-19.—Law of Intersection.

Figure 7-20.—Law of Union.

Whenever you split or join a vinculum, change the sign of operation. That is, AND to OR, or OR to AND.

In applying this theorem it should be remembered that when a vinculum covers part of an expression, the signs under the vinculum change and the signs outside the vinculum do not change; that is,

$$A\bar{B}\bar{C} + \bar{D} + \bar{E} = A(\bar{B} + \bar{C}) + \bar{D}\bar{E}$$

Notice that the grouping of letters must be maintained.

### DISTRIBUTIVE LAW

There are two parts to the distributive law as shown in figure 7-22. The first identity is

$$A(B + C) = AB + AC$$

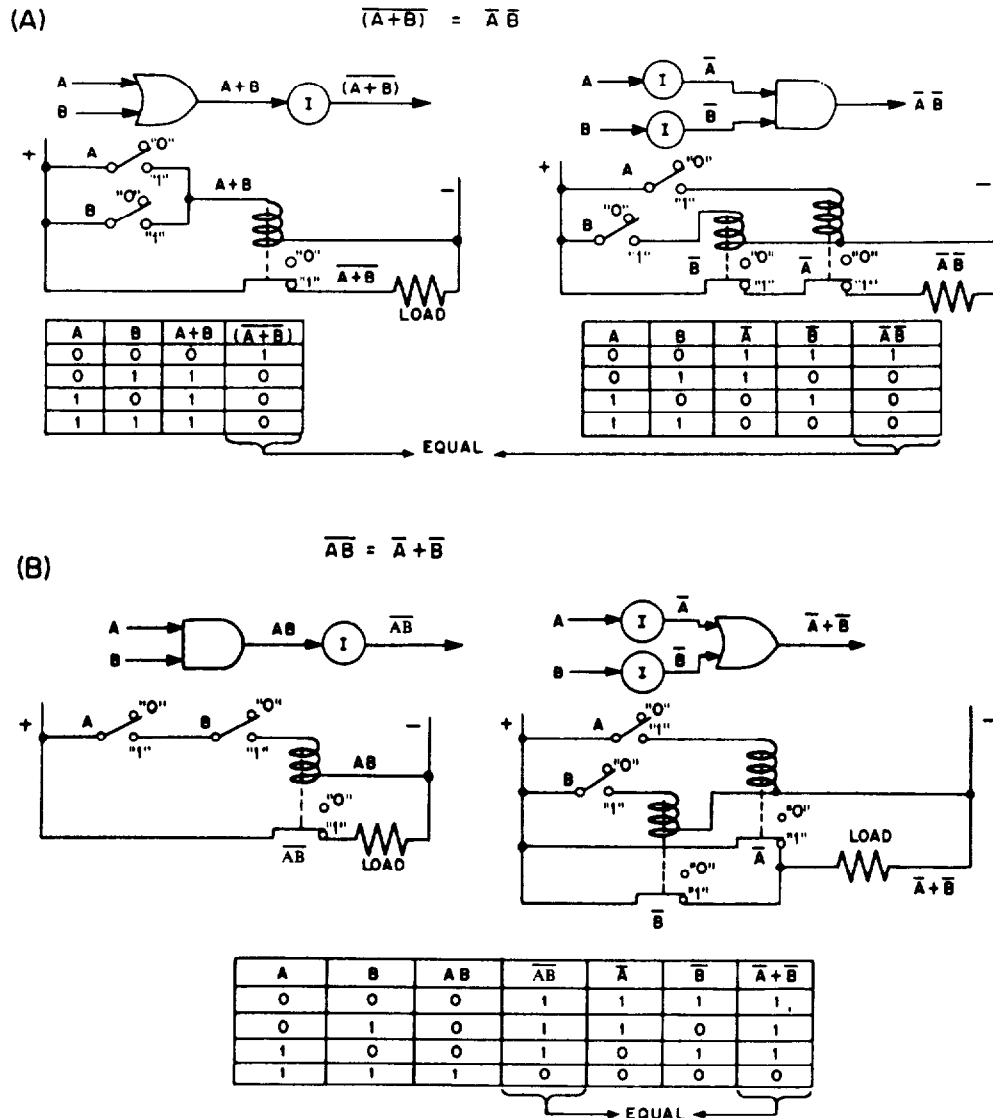


Figure 7-21.—Law of Dualization.

and in order to obtain an output of 1 the A must be 1 and either B or C must be 1. This law is similar to the law of algebra which states that multiplication distributes over addition.

The second identity is

$$A + BC = (A+B)(A+C)$$

and in order to obtain an output of 1, at least one term in each of the parentheses must be 1. THIS LAW DOES NOT APPLY TO ORDINARY

ALGEBRA. If this law did apply to ordinary algebra it would indicate that addition distributes over multiplication. In Boolean algebra this is true. Examples of the distributive law are as follows:

$$A(B+C+D) = AB + AC + AD$$

and

$$A + (B+C)(D+E) = (A+B+C)(A+D+E)$$

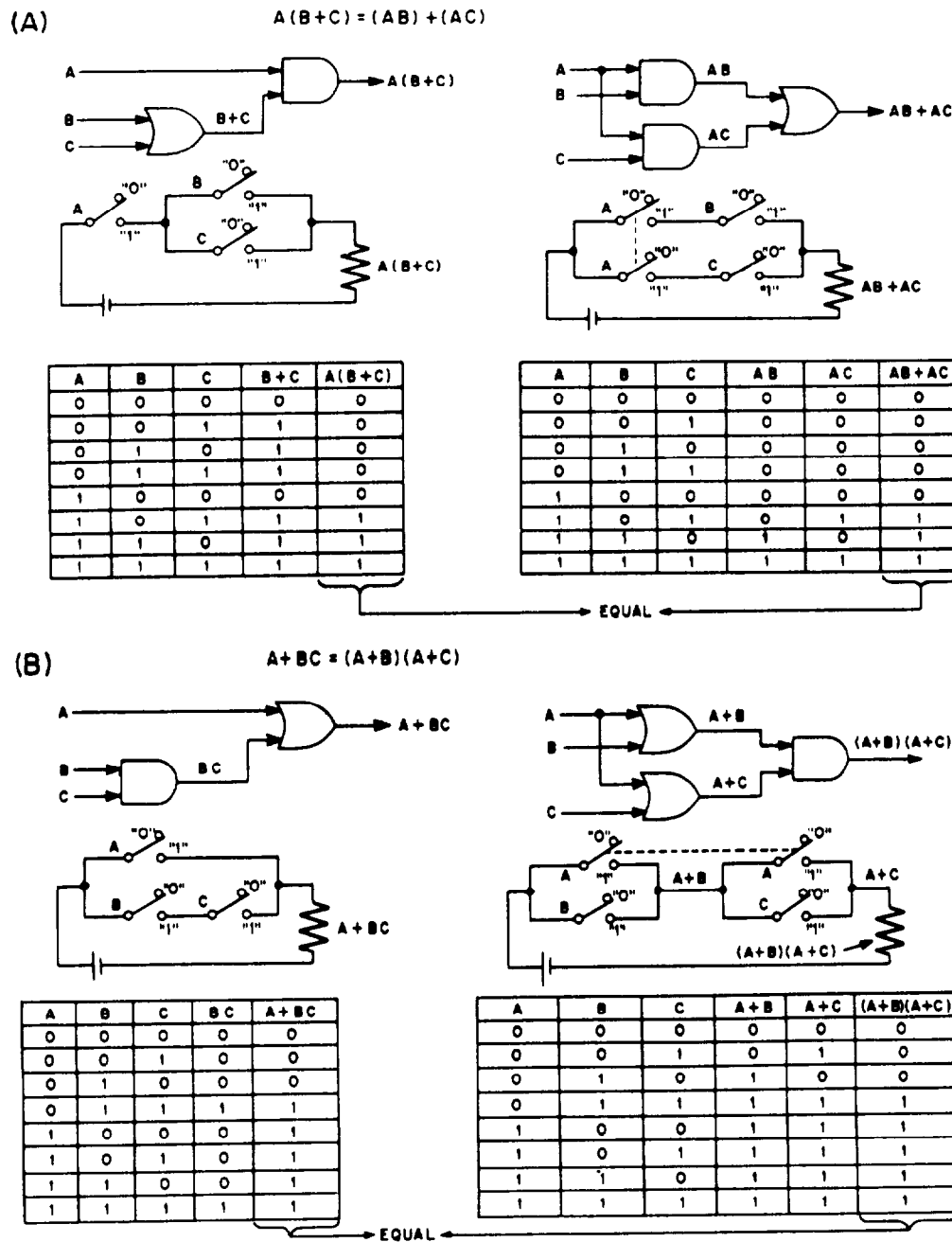


Figure 7-22.—Distributive Law.

# LAW OF ABSORPTION

The law of absorption is shown in figure 7-23. This law is written as

$$A(A + B) = A$$

and

$$A + AB = A$$

and indicates that the output is 1 whenever A is 1. Examples are

$$D(1 + E) = D \cdot 1 = D$$

and

$$A + AB + AC = A(1 + B + C)$$

$$= A \cdot 1$$

$$= A$$

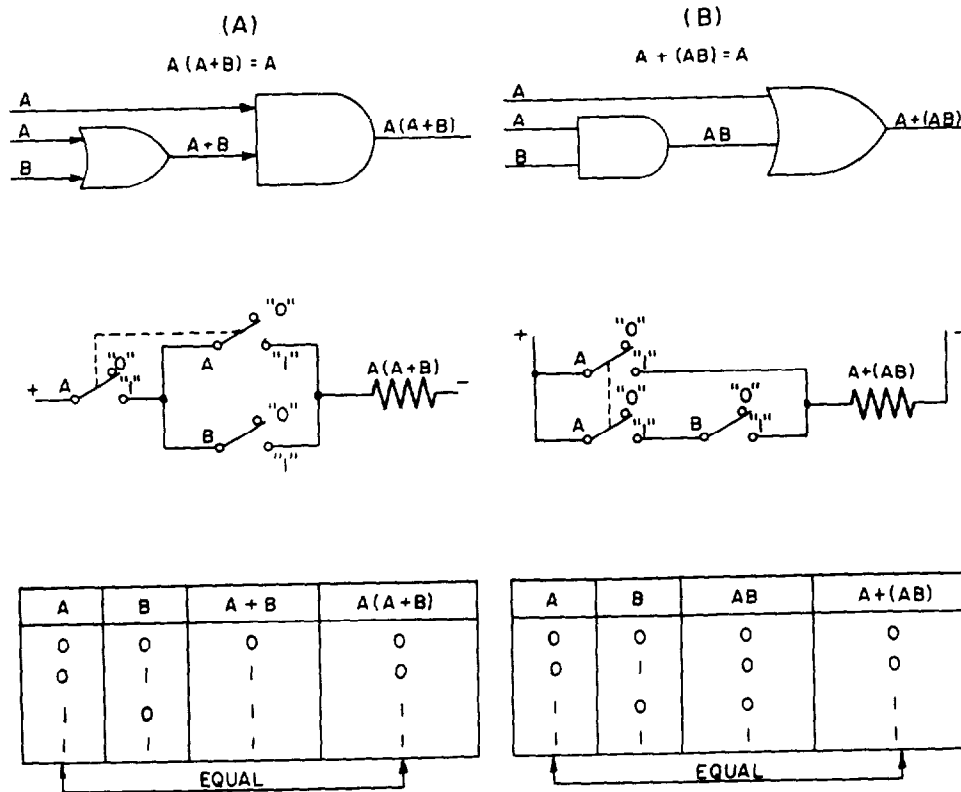


Figure 7-23.—Absorption Law.